Improving the Interaction between Overlay Routing and Traffic Engineering

Gene Moo Lee, Taehwan Choi, and Yin Zhang
Department of Computer Sciences,
The University of Texas at Austin
{gene, ctlight, yzhang}@cs.utexas.edu

Abstract

Overlay routing has been successful as an incremental method to improve the current Internet routing by allowing users to select their logical routing. In the meantime, traffic engineering (TE) techniques are being used to reduce the network cost by adapting the physical routing in response to varying traffic patterns. An overlay is interested in the optimal routes for its own users, whereas TE is to optimize the whole network performance. Previous studies [1, 2] have shown that the conflict of objectives can cause huge network cost increases and oscillations to the network.

In this paper, we improve the interaction between overlay routing and TE by modifying the objectives of both parties. For the overlay part, we propose TE-awareness which limits the selfishness by some bounds so that the action of overlay does not offensively affect TE's optimization process. For the TE part, we suggest COPE as a strong candidate that achieves close-to-optimal performance for predicted traffic matrices and that handles unpredictable overlay traffic efficiently. With extensive simulation results, we show the proposed methods can significantly improve the interaction with lower network cost and smaller oscillation problems.

1 Introduction

Overlay routing has been proposed as an *incremental* method to enhance the current Internet routing without requiring additional functionality from the IP routers. Overlay techniques have been successful for many applications, including application-layer multicast [3–5], web content distribution [6], and overlay routing [7, 8].

In an overlay network, several overlay nodes form an application-layer logical network on top of the IP layer network. Overlay networks enable users to make routing decisions at the application layer by relaying traffic among overlay nodes. We can achieve better route than default IP routing because some problematic and slow links can be bypassed. In addition, overlay routing can take advantage of some fast and reliable paths, which could not be used in the default IP routing due to the business relationship.

By its nature, overlay routing has selfish behavior. In other words, overlay acts strategically to optimize its performance. This nature of overlay makes impact on the related components of the network. Overlay routing has *vertical interaction* with IP layer's traffic engineering. Whenever overlay network changes its logical routing, the physical traffic pattern changes, which is observed by the underlay routing. Network operators use traffic engineering techniques [9–11] to adapt the routing to cope with the new traffic demands. This new routing, in turn, changes the link latency observed by the overlay network, and then overlay makes another decision to change its routing. Traffic Engineering cares about the network as a whole, in order to provide better service to all the users. However, the main objective of overlay routing is to minimize its own traffic latency. Then an interesting issue is to understand the interaction between overlay routing and IP routing.

The interaction between overlay routing and traffic engineering was first addressed by Qiu et al. [1], where the authors investigate the interaction of overlay routing with OSPF and MPLS traffic engineering. Keralapura et al. [12] examine the interaction dynamics between the two layers of control from an ISP's view. Liu et al. [2] formulate the interaction as a two-player game, where overlay attempts to minimize its delay and traffic engineering tries to minimize the network cost. The paper shows that the interaction causes a severe oscillation problem to each player and that both players gain little or nothing as the interaction proceeds.

In this paper, we propose *TE-aware overlay routing*, which takes the objective of underlay routing into account, instead of blindly optimizing its

(i,j), l	physical link
(i',j')	logical link
$cap(l)$	capacity of a physical link l
$v_{st}(l)$	flow of d_{st} on link l
$f_{st}(l)$	fraction of d_{st} on link l
$t(l)$	traffic rate at link l
d_{st}	total TE demand on physical node pair (s,t)
$d_{s't'}$	overlay demand on pair (s',t')
d_{st}^{under}	TE demand due to underlay traffic
$d_{st}^{overlay}$	TE demand due to overlay flow
$P^{(s't')}$	set of logical paths from s' to t'
$\delta_p^{s't'}$	path mapping coefficient
$h_p^{(s't')}$	overlay flow on logical path p

Table 1: Notations for vertical interaction

performance. Moreover, we argue that it is better off for both players if the underlay routing is oblivious to the traffic demands. We suggest COPE [13] as a strong candidate for this purpose.

The paper is organized as follows. First, we formally describe underlying model in Section 2. Section 3 formulates the interaction of overlay routing and traffic engineering as a non-cooperative two-player game. Then various underlay routing schemes are described in Section 4, and selfish overlay routing and TE-aware overlay routing are given in Section 5. Section 6 evaluates the proposed methods with extensive simulation results. Lastly, Section 7 concludes the paper and gives future direction.

2 Model

In this section, we describe the mathematical model, which will be used throughout the paper. Basically, traffic engineering and overlay have different viewpoints of the network. Network operators know all the underlying structure of the physical network, whereas overlay has a logical view of the network.

Table 1 summarizes the notations for vertical interaction. First, we use $G = (V, E)$ to denote an underlay network, where V is the set of physical

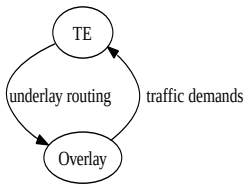


Figure 1: vertical interaction game: TE determines the physical routing, which decides link latency experienced by overlay. Given the observed latency, overlay optimizes its logical routing and changes the physical traffic demands, which, in turn, affects the underlay routing.

nodes and E is the set of edges between nodes. We use l or (i, j) to denote a link and $cap(l)$ to refer the capacity of link l. For the overlay network, we use $G' = (V', E')$. In G', we use i' to represent the overlay node built upon physical node i in underlay graph G. Overlay node i' is connected to j' by a *logical link* (i', j'), which corresponds to a physical *path* from i to j in G.

Now, we need to have different notations for overlay and underlay traffic demands: d_{st} is used to indicate the total traffic demand from node s to t, including overlay and non-overlay traffics, and d_{st} is a sum of d_{st}^{under} and $d_{st}^{overlay}$. d_{st}^{under} refers to the background traffic by non-overlay demands. Next, it is important to differentiate $d_{st}^{overlay}$ from $d_{s't'}$: $d_{s't'}$ indicates the logical traffic demand from overlay node s' to t', whereas $d_{st}^{overlay}$ is the physical traffic demand on physical node pair (s, t), generated by overlay network. In other words, $d_{st}^{overlay}$ is computed by the overlay routing based on the current logical demand $\{d_{s't'} | \forall s', t' \in E'\}$.

The third group of notations is for the overlay routing. $P^{(s't')}$ is the set of logical paths from s' to t'. $\delta_p^{s't'}$ is the path mapping coefficient, where the value is 1, if logical link (s', t') is on logical path p, and 0, otherwise. $h_p^{(s't')}$ is the amount of overlay demand $d(s't')$ flowing on logical path p.

3 Vertical Interaction Game

Based on the formulations in the previous section, traffic engineering and overlay routing are coupled through the mapping from the logical level path to physical level links. We can formulate the interaction as a non-cooperative two-player game as described in Figure 1.

The first player is the ISP's traffic engineering and the second player is

the overlay routing for the user's side. The interaction consists of *sequential* moves of the two players. Each player takes turn and makes action to optimize its performance. Based on the overlay demand and flow conditions on the physical links, overlay calculates the optimal flows on the logical routing. These logical flows and the underlay background traffic are coupled to form the total traffic matrix, which is the input for traffic engineering. Then traffic engineering optimizes its performance by adapting the flows on the physical links, which in turn affects the delays experienced by the overlay. This interaction continues until the two players come up with the Nash equilibrium point [14, 15].

In game theory, Nash equilibrium is a kind of optimal collective strategy in a game involving two or more players, where no player has anything to gain by changing only his or her own strategy. If each player has chosen a strategy and no player can benefit by changing his or her strategy while the other players keep theirs unchanged, then the current set of strategy choices and the corresponding payoffs constitute a Nash equilibrium.

Liu et al. [2] prove the existence of Nash Equilibrium in a simple interaction game, where the topology consists of three nodes and there is a single demand between two nodes. Even though we might prove the existence of a convergent point, the interaction process does not guarantee that two players' behaviors converge to the Nash equilibrium. Moreover, if the game gets complicated, it is even harder to anticipate the interaction process. The authors show that in a realistic scenario, both traffic engineering and overlay routing experience substantial performance loss due to the oscillation.

The main direction of our work is to improve the vertical interaction between overlay routing and traffic engineering. First, we want the interaction game to converge faster because the oscillation in this game degrades the performance of both players. Next, we try to reduce the performance variation in the transient oscillation process.

4 Traffic Engineering

In this section, we now formulate three traffic engineering schemes: Multi-Protocol Label Switching (MPLS) [10], oblivious routing [16], and Convex-hull-based Optimal traffic engineering with Penalty Envelope (COPE) [13]. There are other IP routing protocols not considered in the thesis, such as Open Shortest Path First (OSPF) [9]. We do not consider this method

because it is shown in [1] the vertical interaction of the scheme is inefficient.

The output of traffic engineering is IP-layer routing, which specifies how traffic of each Origin-Destination (OD) pair is routed across the network. Typically, there is path diversity, that is, there are multiple paths for each OD pair, and each path routes a fraction of the traffic.

4.1 Multi-Protocol Label Switching Traffic Engineering

Multi-Protocol Label Switching (MPLS) [10] provides an efficient support of explicit routing, which is the basic mechanism for traffic engineering. Explicit routing allows a particular packet stream to follow a predetermined path rather than a path computed by hop-by-hop destination based routing such as OSPF or IS-IS.

The combination of MPLS technology and its traffic engineering capabilities enable the network operator to adaptively load-balance the traffic demands to optimize the network performance. There are two possible ways to describe the network performance: maximum link utilization and total link latency.

First, network operators sometimes worry about over-loaded links, because these links can be a bottleneck for the whole network performance. A slight change of the traffic pattern may overload the over-utilized links. Therefore, we want to minimize the maximum link utilization.

Given the traffic demand matrix $\{d_{st} | \forall s, t \in V\}$, the goal of MPLS traffic engineering is to choose a physical link flow allocation $\{f_{st}(l) | \forall s, t \in V, \forall l \in E\}$ which minimizes the maximum link utilization. The Linear Program model is given as follows:

$$\begin{aligned} \min \quad & r \\ \text{subject to} \quad & f_{st}(l) \text{ is a routing} \\ & \forall \text{ link } l : \sum_{s,t} f_{st}(l)\ d_{st}/cap(l) \leq r \end{aligned}$$

Here, the first constraint ensures that the given routing satisfies the flow conservation constraints. Basically, for each router, total incoming traffic should be equal to total outgoing traffic. It can be described as the following

equations:

$$\sum_{l|dst(l)=y} f_{st}(l) - \sum_{l|src(l)=y} f_{st}(l) = \begin{cases} 1 & \text{if } y = t, \\ -1 & \text{if } y = s, \\ 0 & \text{otherwise} \end{cases}$$

for each OD pair s, t. Here, $l|dst(l) = y$ indicates all links destined to y, and $l|src(l) = y$ means all links sourced from y.

Secondly, we can use total link latency as the network performance metric, and we want to minimize the total link latency throughout the network. We use the *M/M/1* delay formula to calculate link cost. For a physical link l with capacity $cap(l)$, if its traffic rate is $t(l)$, the total delay experienced by traffic engineering on the links is $\frac{t(l)}{cap(l)-t(l)}$.

Given the traffic demand matrix $\{d_{st}|\forall s, t \in V\}$, the goal of traffic engineering is to choose a physical link flow allocation $\{f_{st}(l)|\forall s, t \in V, \forall l \in E\}$ that minimizes network costs:

$$\begin{aligned} \min \quad & \sum_l \frac{t(l)}{cap(l) - t(l)} \\ \text{subject to} \quad & f_{st}(l) \text{ is a routing} \\ & \forall \text{ link } l : t(l) = \sum_{s,t} f_{st}(l)\ d_{st} \end{aligned}$$

Note that the link latency function is non-linear, which makes the optimization process to be time-consuming. The cost of a link is modeled with a piecewise-linear, increasing, convex function following [17, 18]. We will use this linear function to calculate the latency for overlay routing in Section 5.

4.2 Oblivious Routing

One of the important components of traffic engineering is to understand the traffic flow. Previously discussed MPLS traffic engineering optimizes the paths based on the currently observed traffic matrix. Unfortunately, measuring and predicting traffic demands are really difficult problems. Flow measurements are rarely available on all links and Ingress/Egress points. Moreover, demands change over time on special eventssuch as DoS attac, flash crowds, and internal/external network failures. It seems that the most

one can hope is some approximate picture of demands, not necessarily the very current one.

Oblivious routing [16] is proposed to resolve this issue. It calculates an optimal routing which performs reasonably well *independently* of traffic demands. In other words, this "demand oblivious" routing is designed with little knowledge of the traffic matrix (TM), taking only the topology along with link capacities into account.

4.3 Convex-Hull-Based Optimal Traffic Engineering with Penalty Envelope

MPLS Traffic Engineering can be regarded as an extreme case of online adaptation. An advantage of this scheme is that it achieves the best performance for the current traffic demand. However, if there are significantly fast traffic changes, such method can suffer a large transient penalty. Oblivious routing is a way to handle unpredicted traffic spikes. However, a potential drawback of completely oblivious routing is its sub-optimal performance for the normal traffic demand.

Convex-hull-based Optimal traffic engineering with Penalty Envelope (COPE) [13] is proposed as a hybrid combination of predication-based optimal routing and oblivious routing. COPE handles both dynamic traffic and dynamic inter-domain routes and, at the same time, achieves close-to-optimal performance for normal, predicted traffic matrices.

COPE optimization can be obtained by adding penalty envelope constraints to MPLS Linear Programming. The penalty envelope constraint restricts that the routing f has maximum performance ratio less than or equal to $\overline{r}$. This can be formalized as the following set of linear constraints:

$$\forall \text{ link } l : \sum_m cap(m)\ \overline{\pi}(l,m) \leq \overline{r}$$
$$\forall \text{ link } l, \forall \text{ pair } s \to t : f_{st}(l)/cap(l) \leq \overline{p}_l(s,t)$$
$$\forall \text{ link } l, \forall \text{ node } s, \forall \text{ edge } e = t \to v :$$
$$\overline{\pi}(l, \text{link-of}(e)) + \overline{p}_l(s,t) - \overline{p}_l(s,v) \geq 0$$
$$\forall \text{ link } l, m : \overline{\pi}(l,m) \geq 0$$
$$\forall \text{ link } l, \forall \text{ node } s : \overline{p}_l(s,s) = 0$$
$$\forall \text{ link } l, \forall \text{ node } s,t : \overline{p}_l(s,t) \geq 0$$

The convex-hull-based Linear Program takes the input as a set of possible traffic matrices. In our simulation, however, we use a single currently observed traffic matrix. Still, COPE is shown to make an excellent performance for the dynamic change of the traffic patterns.

5 Overlay Routing

In this section, we formulate the objective functions for overlay routing. We start with the default overlay routing, which we term *selfish overlay routing*. Given the current underlay routing and experienced link latency, selfish overlay tries to minimize its total latency by changing the loads for each logical path in the overlay network.

Then, we propose a variation of overlay routing. Basically, we include additional constraints to the original overlay optimization so that overlay takes the presence of traffic engineering into account. We term this as *TE-aware overlay routing*.

5.1 Selfish Overlay Routing

The overlay routing algorithm determines a logical path flow allocation $\{h_p^{s't'} | \forall s', t' \in V', \forall p \in P^{(s't')}\}$ that minimizes the average delay experienced by the overlay users, whereas traffic engineering determines the physical flow. By $h_p^{s't'}$, we denote the logical overlay demand from s' to t' allocated to path p.

Individual overlay users may choose their routes independently by probing the underlay network. However, we assume that a centralized entity calculates routes for all overlay users. Given the physical network topology, underlay routing, and experienced latency for each link, optimal overlay routing can be obtained by solving the following non-linear optimization problem:

$$
\begin{aligned}
\min \quad & \sum_l \frac{t(l)^{overlay}}{cap(l) - t(l)} \\
\text{subject to} \quad & h_p^{s't'} \text{ is a logical routing} \\
& \forall \text{ link } l : t(l) = \sum_{s,t} f_{st}(l)(d_{st}^{under} + d_{st}^{overlay}) \\
& \forall \text{ link } l : t(l)^{overlay} = \sum_{s,t} d_{st}^{overlay} f_{st}(l)
\end{aligned}
$$

$$\forall s,t \in V : d_{st}^{overlay} = \sum_{s',t',p} \delta_p^{s't'} h_p^{(s't')}$$

The first constraint ensures that the logical routing satisfies the logical flow conservation constraints. This can be expressed as follows:

$$\forall s',t' \in V' : \sum_{p \in P^{(s't')}} h_p^{(s't')} = d^{(s't')}$$
$$\forall s',t' \in V' : h_p^{(s't')} \geq 0.$$

Note that the main objective of problem is non-linear. But we can again linearize the non-linear part of the program by using the same technique used for the traffic engineering optimization. Refer to the Section 4 for details.

5.2 TE-Aware Overlay Routing

Based on the selfish overlay routing, we can include additional constraints to ensure the overlay is *TE-aware*. By TE-awareness, we mean the selfishness of the overlay is limited by some bound so that the action of overlay does not offensively affect the traffic engineering's optimization process.

The basic idea is this: (1) when the current latency is below the average latency, the overlay tries to minimize its own traffic amount, given that the current latency is preserved (load-balancer). (2) If the latency is above the average, then overlay changes the logical routing to improve the latency, but, at the same time, it avoids a specific link to be overloaded (limited-optimizer).

The first part, load-balancer, can be formalized as the following Linear Program model:

$$\begin{aligned}
\min \quad & \sum_{s,t} d_{s,t}^{overlay} \\
\text{subject to} \quad & h_p^{s't'} \text{ is a logical routing} \\
& \forall \text{ link } l : t(l) = \sum_{s,t} f_{st}(l)(d_{st}^{under} + d_{st}^{overlay}) \\
& \forall \text{ link } l : t(l)^{overlay} = \sum_{s,t} d_{st}^{overlay} f_{st}(l) \\
& \forall s,t \in V : d_{s,t}^{overlay} = \sum_{s',t',p} \delta_p^{s't'} h_p^{(s't')}
\end{aligned}$$

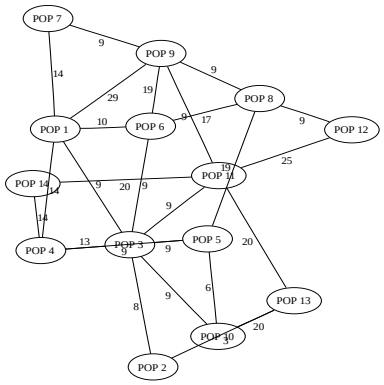


Figure 2: 14-node Tier-1 backbone topology: each node represents a Point-of-Presence (POP) and each link represents the aggregated connectivity between the routers belonging to a pair of adjacent POPs. Four POPs (3,6,7,11) are used as overlay nodes.

$$\sum_{l} \frac{t(l)^{overlay}}{cap(l) - t(l)} \leq \Theta$$

Here, the main objective is to minimize the total overlay traffic amount. The last constraint guarantees that the current latency is preserved. (Θ in the last constraint indicates the current latency.)

Secondly, limited selfish overlay routing can be defined by adding a constraint to the default selfish overlay routing:

$$\forall \text{ link } l : \sum_{s,t} f_{st}(l)\ d_{st}^{overlay} \leq \theta$$
$$\theta = \max\{t(l)^{overlay} | \forall \text{ link } l\}$$

Here, θ is the maximum link load that the overlay generates in the past run. The additional contraint limits the selfishness of overlay and prevents specific links to be overloaded.

6 Simulation

This section describes the simulation results of vertical interactions. We first compare MPLS and COPE as the underlay traffic engineering schemes.

Then we evaluate and compare TE-aware overlay routing and selfish overlay routing.

6.1 Data Set Description

We perform extensive experiments on a 14-node Tier-1 POP topology described in [19]. The underlay network topology is given in Figure 2. We have done experiments with other topologies and observed qualitatively consistent results. On top of the physical network, we made up a four-node full-meshed overlay network of node 3, 6, 7, and 11. For the traffic matrix, we generate synthetic traffic demands using gravity model [19].

6.2 Implementation

We use General Algebraic Modeling System [20] to implement various optimization procedures for the experiments. Then the interaction between optimization programs is implemented by connecting the inputs and outputs of the GAMS programs through Perl scripts. Given that we run the optimization process for more than hundred iterations, we need a support of Condor [21], which is a specialized workload management system for compute-intensive jobs.

6.3 MPLS and COPE with Selfish Overlay

We start with the comparison between MPLS and COPE in the operator's viewpoint. We fix the overlay routing to be selfish and compare the performance of MPLS and COPE.

For the COPE, we need a prespecified penalty envelope value. We first calculate the value (1.9969) by running oblivious routing, which finds the optimal routing which minimizes the oblivious ratio. This can be calculated without any information about traffic demands because oblivious routing only depends on the network topology information. Then by multiplying 1.1 to the optimal oblivious ratio, we set the penalty envelope value.

First, we set 10% of the total traffic demand to be operated by the selfish overlay routing. We set the load scale factor to be 0.3, 0.5, and 0.7. This means that the maximum link utilization is 30%, 50%, and 70%, respectively, when all the demands use the default underlay routing without overlay's action.

The experiment results are shown in Figure 3. Regardless of the load scale factor, we can observe that the COPE makes better interaction with selfish overlay. In all cases, MPLS traffic engineering suffers from substantially large oscillation throughout the interaction, where COPE achieves almost stable performance with its maximum link utilization. Similarly, the dynamics of overlay latency is quite stable with the interaction of COPE. Moreover, the average latency sometimes gets improved by using COPE.

For the next experiment, we want to explore the impact of the overlay fraction to the vertical interaction. Now, we fix the load scale factor and change the fraction of overlay traffic (10%, 30%, 50%) in the experiment. We set the load scale factor to be 0.9 in Figure 4.

Again, we can observe that COPE makes better interaction with selfish overlay routing than MPLS does. As we increase the fraction of overlay traffic, the oscillation of maximum link utilization gets larger, which follows our intuition. However, the performance of overlay latency seems to be independent of how much portion overlay routing operates.

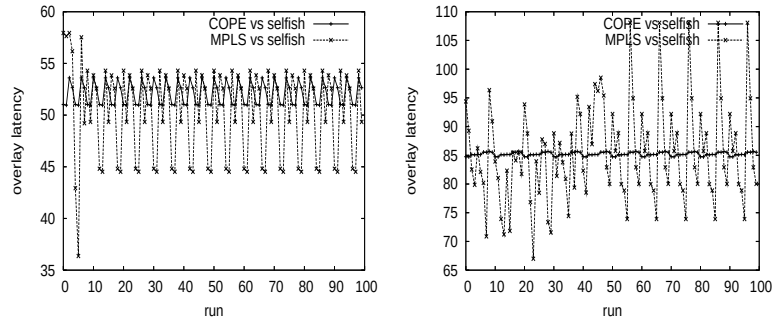
With the extensive simulation, we find COPE as a strong traffic engineering technique which achieves stable performance even the selfish overlay traffic dominates a significant portion of the total traffic demand.

6.4 TE-Aware Overlay and Selfish Overlay with MPLS

Now, we evaluate the TE-aware overlay routing by comparing it to the selfish overlay routing. For the underlay routing, we again use MPLS and COPE. We first start the evaluation by comparing two overlay routings on top of MPLS traffic engineering.

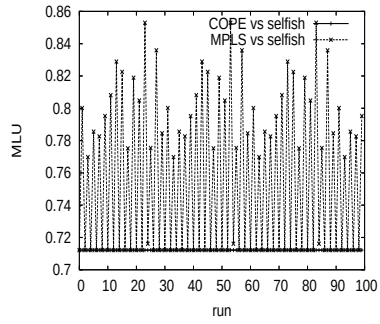
In Figure 5, we set 10% of the traffic to be operated by overlay routing and increase the load scale factor (0.3, 0.5, 0.7). Considering the overlay latency, TE-aware overlay routing achieves more stable performance. Moreover, in the case where the load scale factor is 0.5 and 0.7, the average latency experienced by TE-aware overlay is lower than selfish overlay. We can see that overlay routing can achieve better and stable routing by understanding the objective of underlay routing.

Considering the traffic engineering side, selfish overlay routing makes significant burden to the underlay routing because it generates substantially large amount of additional traffic. Thus, we can observe sudden increase of the maximum link utilization in all cases. However, TE-aware overlay limits its selfishness and tries to avoid a specific link to be over-loaded by its own

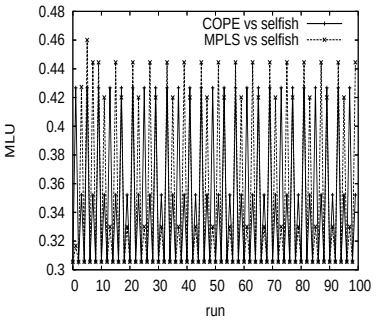


(a) Load Scale Factor 0.3

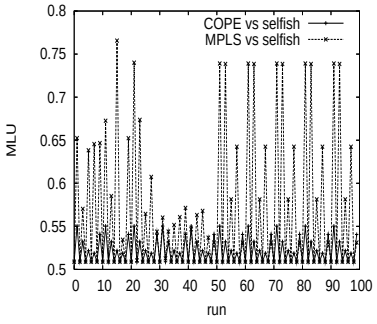
(b) Load Scale Factor 0.5



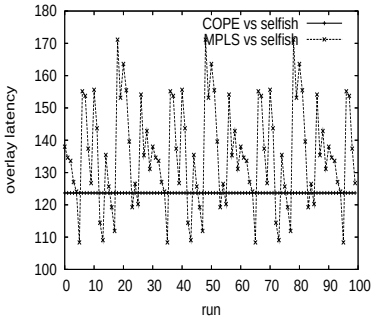
(c) Load Scale Factor 0.7



(d) Load Scale Factor 0.3

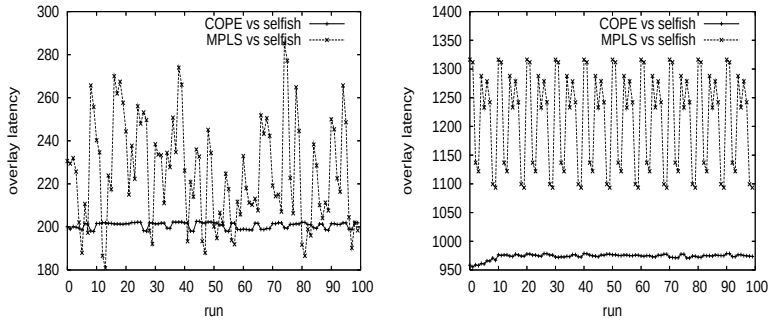


(e) Load Scale Factor 0.5



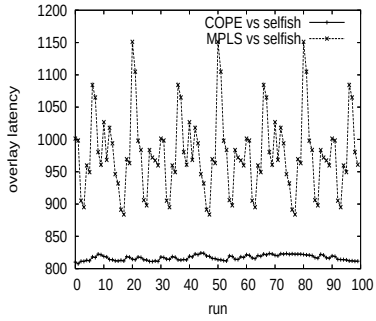
(f) Load Scale Factor 0.7

Figure 3: MPLS and COPE with selfish overlay. 14-node topology with a 4-node overlay network, overlay fraction = 10%.

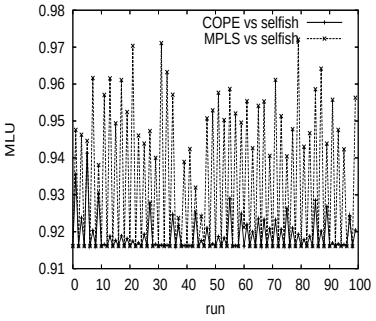


(a) Overlay Fraction 10%

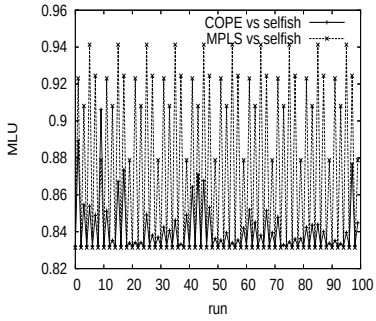
(b) Overlay Fraction 30%



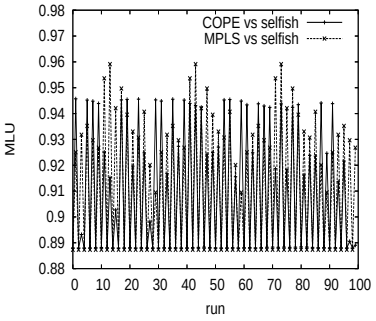
(c) Overlay Fraction 50%



(d) Overlay Fraction 10%



(e) Overlay Fraction 30%



(f) Overlay Fraction 50%

Figure 4: MPLS and COPE with selfish overlay. 14-node topology with a 4-node overlay network,load scale factor = 0.9.

traffic. Thus, the fluctuation of maximum link utilization is smaller when the overlay is TE-aware.

Next, we fix the load scale factor to be 0.9, and change the overlay fraction: 10%, 30%, and 50% (Figure 6). The experiments are conducted where the network is substantially congested (90% - 120%). Still, the proposed method makes better interaction than selfish overlay does.

With the extensive experiment results, we come up with the conclusion that TE-aware overlay routing generally makes stable interaction with MPLS traffic engineering. Selfish overlay routing experiences less predictable latency and it makes significantly large maximum link utilization of the network. However, we can achieve either convergence or regular pattern with the overlay latency when TE-awareness is used in overlay routing. Moreover, the network overhead to the traffic engineering is reduced by using the proposed overlay routing. Thus, TE-awareness obtains win-win game for each player in the presence of MPLS traffic engineering.

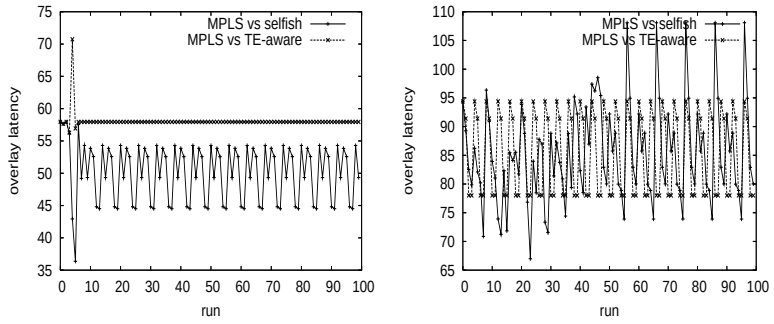
6.5 TE-Aware Overlay and Selfish Overlay with COPE

For the last experiment, we examine the interaction between TE-aware overlay and selfish overlay on top of the COPE traffic engineering. In the previous experiments comparing COPE and MPLS, we have observed that COPE achieves better interaction with selfish overlay routing. Now, the question is how much gain we can get by using TE-aware overlay with COPE.

Figure 7 describes the experiment results, where 10% of the traffic is routed by overlay routing. We again use three load scale factors (0.3, 0.5, 0.7). Different from the experiments with MPLS, the achievement we get from TE-awareness is limited. When the load scale factor is 0.3 and 0.5, the TE-aware overlay routing converges fast with good latency, but the proposed method shows a small oscillation in the last case. However, comparing to the oscillation in MPLS experiments, we can see the performance variation is negligible. Similar patterns can be observed with the maximum link utilization.

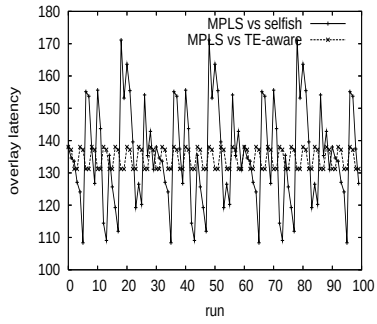
In Figure 8, we fix the load scale factors to be 0.9 and change the fraction of overlay traffic (10%, 30%, 50%). General observation is that as the link gets more utilized, the performance gain from TE-awareness is substantially large.

Considering the overlay side, in all experiments, the latency experienced by TE-aware overlay is better than that of selfish overlay. The maximum link

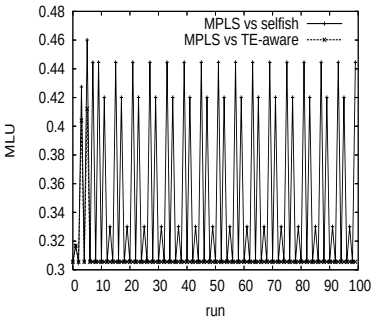


(a) Load Scale Factor 0.3

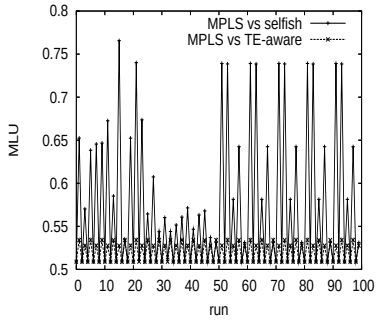
(b) Load Scale Factor 0.5



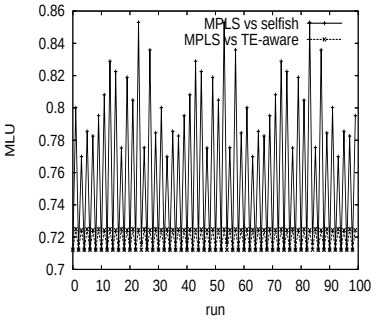
(c) Load Scale Factor 0.7



(d) Load Scale Factor 0.3

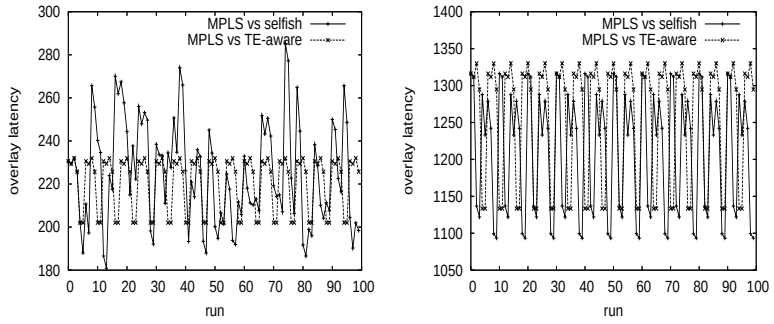


(e) Load Scale Factor 0.5



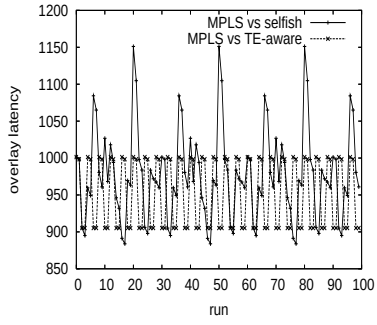
(f) Load Scale Factor 0.7

Figure 5: TE-aware overlay and selfish overlay on MPLS. 14-node topology with a 4-node overlay network, overlay fraction = 10%.

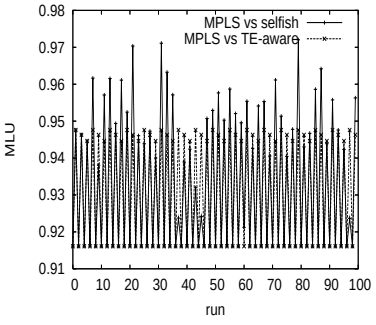


(a) Overlay Fraction 10%

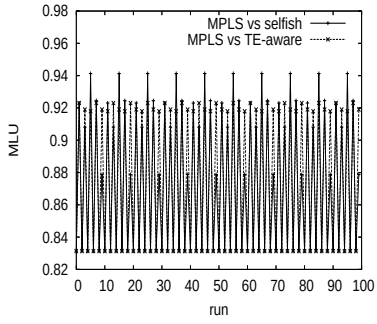
(b) Overlay Fraction 30%



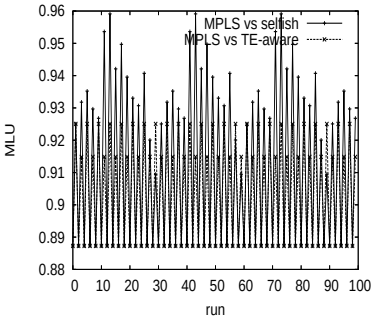
(c) Overlay Fraction 50%



(d) Overlay Fraction 10%

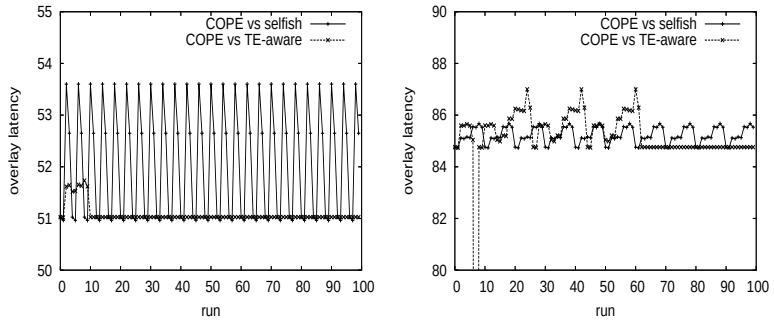


(e) Overlay Fraction 30%



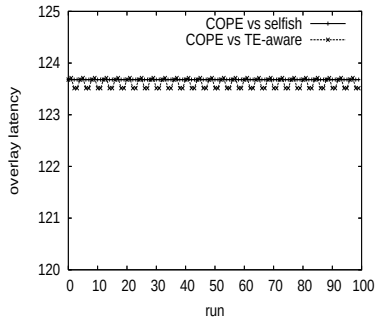
(f) Overlay Fraction 50%

Figure 6: TE-aware overlay and selfish overlay on MPLS. 14-node topology with a 4-node overlay network,load scale factor = 0.9.

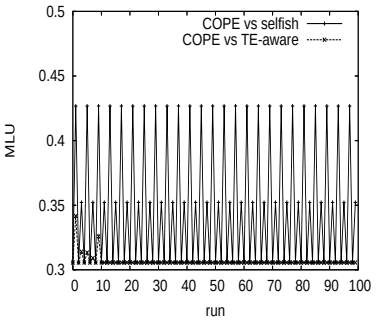


(a) Load Scale Factor 0.3

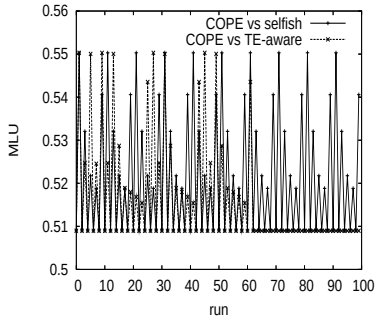
(b) Load Scale Factor 0.5



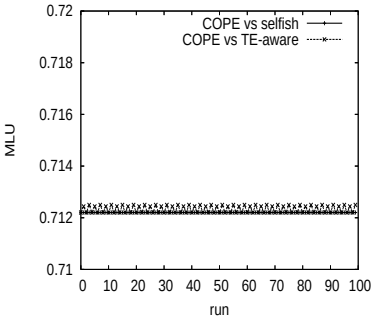
(c) Load Scale Factor 0.7



(d) Load Scale Factor 0.3



(e) Load Scale Factor 0.5



(f) Load Scale Factor 0.7

Figure 7: TE-aware overlay and selfish overlay on COPE. 14-node topology with a 4-node overlay network,overlay fraction = 10%.

latency experienced by selfish overlay is sometimes twice larger than average latency. In some scenario, the selfish overlay latency keeps increasing as the interaction with underlay proceeds. TE-aware overlay shows similar pattern but makes convergence at considerably lower latency. Looking at the traffic engineering side, TE-awareness obtains either similar or better performance than selfishness.

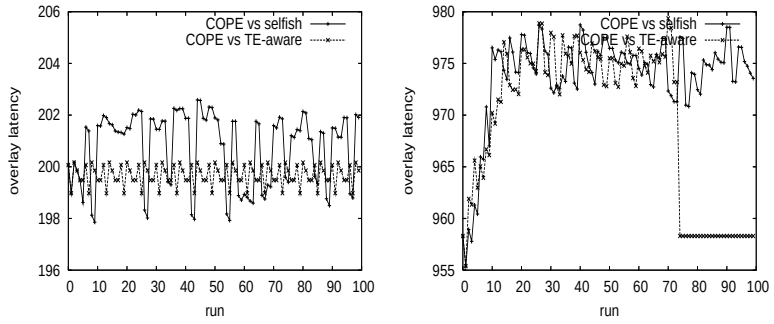
Summarizing the interaction experiments with COPE, we can achieve considerably good interaction for both selfish overlay and TE-aware overlay. But, TE-aware overlay performs slightly better than selfish overlay routing.

7 Conclusion and Future Direction

In this paper, we improve the *vertical interaction* between overlay routing and traffic engineering by modifying the objectives of both parties. Overlay changes the physical traffic demands by the logical routing decisions, and dynamically changing traffic demands affect the performance of underlay routing. Specifically, network operators use traffic engineering techniques to adapt IP layer routing to cope with the new traffic matrix. We propose *TE-aware overlay routing*, which takes traffic engineering's objective into account in overlay routing decision. Then we also suggest COPE as a strong traffic engineering technique, which makes a good interaction with the unpredictable overlay traffic demands. We show the feasibility of the proposed methods with extensive simulation results.

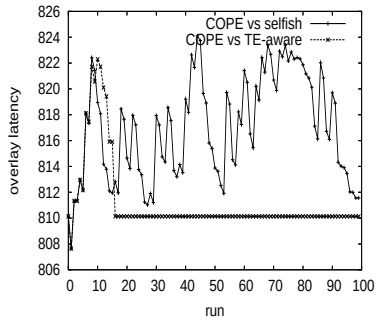
So far, our model only captures interactions within a single domain. Then a future direction will be to extend the arguments to inter-domain level. In this scenario, overlay nodes are spread across several ASes and cooperate each other. Then the action of overlays will make interaction with inter-doman routing algorithms. Another direction is to explore the vertical interaction in more realistic scenario. For example, we do not consider link failures in the scenario. It will be interesting to see the interactions after a specific link failed.

Lastly, we use average latency as an indicator of decision for overlay. In reality, this may not be available. We may use scheme similar to TCP congestion control mechanism. In other words, overlay additively increases the selfishness until some congestion, and exponentially back off to ease the congestion level. Other possibility is to consider *oblivious overlay routing*, which guarantees overlay performance for every possible underlay routing.

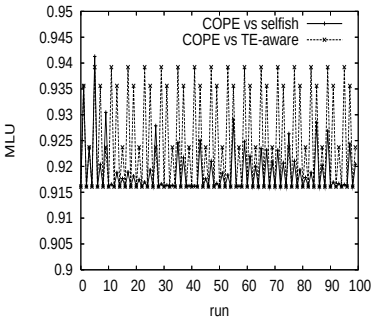


(a) Overlay Fraction 10%

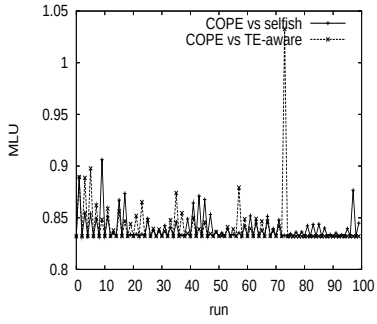
(b) Overlay Fraction 30%



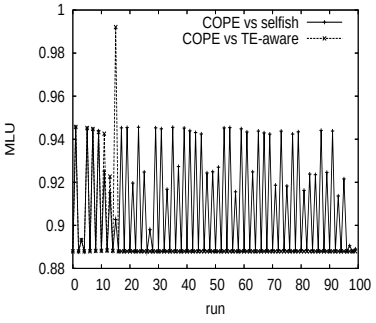
(c) Overlay Fraction 50%



(d) Overlay Fraction 10%



(e) Overlay Fraction 30%



(f) Overlay Fraction 50%

Figure 8: TE-aware overlay and selfish overlay on COPE. 14-node topology with a 4-node overlay network,load scale factor = 0.9.

References

[1] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "On selfish routing in Internet-like environments," in *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, august 2003, pp. 151–162.

[2] Y. Liu, H. Zhang, W. Gong, and D. Towsley, "On the interaction between overlay routing and traffic engineering," in *IEEE Conference on Computer Communications (INFOCOM)*, 2005.

[3] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay muilticast architecture," in *Proceedings of ACM SIGCOMM*, 2001, pp. 55–67.

[4] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. O. Jr, "Overcast: Reliable multicasting with an overlay network," in *Proceedings of Symposium on Operating Systems Design and Implementation (OSDI)*, 2000, pp. 197–212.

[5] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "OverQoS: An overlay based architecture for enhancing internet QoS," in *Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI)*, 2004, pp. 71–84.

[6] "Akamai Technologies, Inc. http://www.akamai.com."

[7] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of Symposium on Operating Systems Principles (SOSP)*, 2001, pp. 131–145.

[8] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan, "Detour: a case for informed internet routing and transport, Tech. Rep. TR-98-10-05, 1998.

[9] "OSPFv3 overview," 2003.

[10] J. Ruela and M. Ricardo, "MPLS - Multi-Protocol Label Switching," in *The Industrial Information Technology Handbook*, 2005, pp. 1–9.

[11] J. W. Stewart, III, *BGP4: inter-domain routing in the Internet.* Addison-Wesley, 1999.

[12] R. Keralapura, N. Taft, C.-N. Chuah, and G. Iannacco, "Can ISPs take the heat from overlay networks?" in *Proceedings of the Third Workshop on Hot Topics in Networks*, November 2004.

[13] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "COPE: Traffic engineering in dynamic networks," in *Proceedings of ACM SIGCOMM*, 2006.

[14] P. K. Dutta, "Strategies and games: Theory and practice," 1999.

[15] J. Nash, "Non-cooperative games," *The Annals of Mathematics*, pp. 286–295, September 1951.

[16] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain trafficc demands: Understanding fundamental tradeoffs," in *Proceedings of ACM SIGCOMM*, 2003, pp. 313–324.

[17] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Communications Magazine*, pp. 118–124, October 2002.

[18] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proceedings of IEEE International Conference of Computer Communications (INFOCOM)*, 2000, pp. 519–528.

[19] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic matrix estimation: existing techniques and new directions," in *Proceedings of ACM SIGCOMM*, 2002, pp. 161–174.

[20] "General Algebraic Modeling System. http://www.gams.com."

[21] "Condor. http://www.cs.wisc.edu/condor."